IJARSCT

ISSN (Online) 2581-9429

Impact Factor: 6.252

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

Volume 2, Issue 3, April 2022

# Microservices: Architecture and Technologies

**Aarti Bobhate**
Department of Information Technology
S. S. & L. S. Patkar College of Arts & Science & V. P. Varde College of Commerce & Economics, Mumbai
bobhateaarti9@gmail.com

**Abstract:** *A microservices framework consisting of microservices and containers delivers a diverse and distributed application. This architecture handles complex tasks as a collection of smaller services as opposed to the conventional monolithic architecture. This is made manageable by adopting microservice architecture. Furthermore, it provides scalability, easy maintenance and independent deployment of services to an application. It enables small parts of an application to be updated, replaced and scaled effortlessly.*

**Keywords:** Microservices architecture, monolithic architecture, microservices technology, Docker.

## I. Introduction

The micro services architecture is a development methodology wherein you can fragment a single application into a series of smaller services, each executing in its own process and interacting with lightweight mechanisms. Microservices (or microservices architecture) are a cloud native architectural approach in which a single application is composed of many loosely coupled and independently deployable smaller components, or services.

### 1.1 Monolithic Architecture

A monolithic architecture is the traditional unified model for the design of a software program. Monolithic, means composed all in one piece. The adjective monolithic also means both too large and unable to be changed.

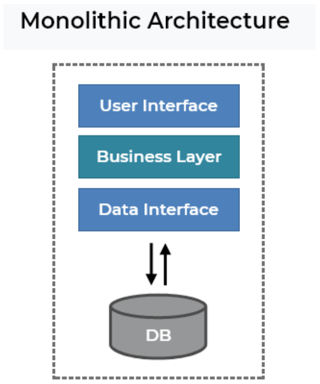


**Figure 1:** Monolithic Architecture

Figure1 shows the monolithic architecture. Components of a monolithic application cannot be deployed individually, conspicuously on every update the entire application needs to be redeployed. Monolithic applications are usually divided into modules or layers like presentation, business, database, application, etc. Even though, this type of architecture is simple to develop it has many drawbacks. These limitations are overcome by the microservice architecture. Some limitations of monolithic architecture are listed below.

1. Every small change in application requires complete redeployment of entire application.
2. Difficult to manage due to limitation in size and complexity.
3. It is difficult to use new technologies.
4. Limitations in scaling.
5. A single defect affects the entire application.

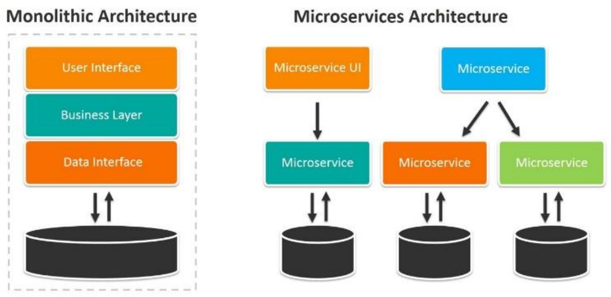
## II. Comparative Analysis of Microservices Architecture and Monolithic Architecture

### 2.1 Microservices Architecture

1. Highly scalable with the use of containers and cloud.
2. Rapid Continuous Deployment. Each service can be deployed independently.
3. Can use diverse technology stack as it is loosely coupled with high cohesion.
4. A single bug affects only the service in which it occurs.

### 2.2 Monolithic Architecture

1. It is difficult to scale monolithic application.
2. Slow deployment. All updates require redeployment of entire application.
3. Uses a single technology stack as it is tightly coupled with low cohesion.
4. A single bug can affect the entire application.



### 2.3 Advantages of Microservices Architecture

1. Microservices are independently deployable and allow for more team autonomy
   a. Each microservice can be deployed independently, as needed, enabling continuous improvement and faster app updates.
2. Microservices are independently scalable.
   a. As demand for an app increase, it’s easier to scale using microservices. You can increase resources to the most needed microservices rather than scaling an entire app. This also means scaling is faster and often more cost-efficient as well.
3. Microservices reduce downtime through fault isolation.
   a. If a specific microservice fails, you can isolate that failure to that single service and prevent cascading failures that would cause the app to crash. This fault isolation means that your critical application can stay up and running even when one of its modules fails.
   b. The smaller codebase enables teams to more easily understand the code, making it simpler to maintain.

### 2.4 Disadvantages of Microservices Architecture

1. Microservices create different types of complexity than monolithic applications for development teams.
   a. First, communication between services can be complex. An application can include dozens or even hundreds of different services, and they all need to communicate securely.
   b. Second, debugging becomes more challenging with microservices. With an application consisting of multiple microservices and with each microservice having its own set of logs, tracing the source of the problem can be difficult.
   c. And third, while unit testing may be easier with microservices, integration testing is not. The components are distributed, and developers can’t test an entire system from their individual machines.
2. Interface control is even more critical
   a. A microservices architecture model results in a large number of APIs, all crucial to the operation of the enterprise — so interface control becomes mission-critical.

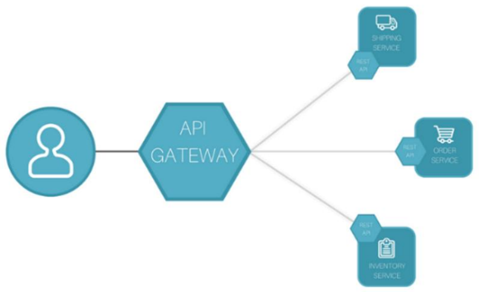
3. Up-front costs may be higher with microservices.
    a. For microservices architecture to work for your organization, you need sufficient hosting infrastructure with security and maintenance support, and you need skilled development teams who understand and manage all the services.

### 2.5 Benefits of Microservices Architecture

1. **Isolation:** Microservices are profitable due to their isolation and resilience. If one of the components fail, developers have the option to use another service and the application will continue to run independently. This way, engineers can build and deploy services without the need to change the whole app.
2. **Scalability:** It's easier for development teams to scale up or down following the requirements of a specific element. Isolation allows apps to run correctly when massive changes are happening. Microservices prove to be the perfect approach for companies working with various platforms and devices.
3. **Productivity:** Microservice architecture is the ability to easily understand when compared to an entire monolithic app. If you plan to expand your development team, microservices are a better pick.
4. **Flexibility:** The microservice approach lets developers choose the right tools for the right task.
5. **Faster Project Development:** Microservices work independently, so you don't have to change the codebase in order to modify the features. You can change one component, test, and then deploy it individually. In turn, you will deliver the app faster.

## III. Microservices Architecture and Components

1. **Client:** Clients start of the architecture by requesting services from different sources to perform varied tasks.
2. **Identity Provider:** Client requests need to be authenticated and authorized. It is important for all services to be secure so as to not be exploited. Every service needs to be identified on service call with the role defined so that the caller is authorized. Identity providers are used to store and verify user ID for communication between client and system. The requests are then passed forward to API Gateways to connect the client to the services.
3. **API Gateway:** An Application Program Interface Gateway in a microservice architecture provides singular entrance to the system. It is capable of creating multiple custom API for a microservice depending on the platform being used and its requirements.



4. **Database:** In microservices architecture each service has its own database to store data. Each service uses its private database to complete its service specific task.
5. **Containers:** Containers need not necessarily be a part of microservice architecture but significantly ameliorate speed and efficiency. They are packages consisting of a singular part of system i.e. microservices. Also, they are not allowed access to rest of the architecture, but only microservices and its dependencies. Containers are independent like microservices, thereby aiding in scaling.
6. **Service Mesh:** Service mesh is used for secure inter-communication between different microservices in a distributed application. For smooth and safe communication, service mesh residing in the infrastructure layer help in networking by managing, observing and securing services.
   Service mesh are as follows.
   1. Traffic Management

2. Access Control
3. Routing
4. Load balancing

**3.1 Service Discovery**

Service discovery is the process of locating existing services that are relevant for a given request based on the description of their functional and non-functional semantics. Service discovery is made up of.

1. Service directory
2. Service provider
3. Service consumer

**3.2 Implementation Tools**

**A. Programming Languages**

1. To ensure time conservation, it is important that one selects the language which has less code to type. So, C should be avoided.
2. Choose a language that gives you speed and can control traffic.

**3.3 Top languages to use for microservices are as follows**

- **Python:** Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.
- **Java:** Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

**3.4 Technologies**

- **Docker:** Docker is an open-source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies requiredto run that code in any environment. Containers simplify the delivery of distributed applications and have become increasingly popular as organizations shift to cloud-native development and hybrid multi-cloud environments.
- **Consul:** Consul ensures communication between microservices. Consul is better than other services discovery solutions concerning features like:
  - HTTP Rest API and support for DNS
  - It can auto generate configuration files.

## IV. Conclusion

Microservices architecture serves as an alternative to monolithic architecture. In monolithic architecture, the entire application is developed using a single code base for all components. This architecture has many limitations. Thereby, we adopt the microservices architecture which offers scalability, flexibility, resilience and autonomy.

## References

**[1].** Microservice Architecture
**[2].** Monolithic microservice architecture
**[3].** Best Technologies to build Microservices Architecture